References: 1. *W. B. Arthur.* Increasing Returns and Path Dependence in the Economy. University of Michigan Press, 1994. **2.** *A. L. Barabási, E. Bonabeau.* Scale-free networks. Scientific American 288, pp.60-69, 2003. **3.** *S. Bikhchandani, D. Hirshleifer, I. Welch.* Learning from the Behavior of Others: Conformity, Fads, and Information Cascades. American Economic Association in Journal of Economic Perspectives, pp.151-170, 1998. **4.** *E. Bonabeau.* The Perils of the Imitation Age. Harvard Business Review Article, Jun 1, pp.45-47, 49-54, 2004. **5.** *D. Boyd, N. B. Ellison.* Social Network Sites: Definition, History, and Scholarship. Journal of Computer-Mediated Communication, Vol. 13, No. 1-2, 2007. **6.** *J. Breslin, S. Decker.* The Future of Social Networks on the Internet – The Need for Semantics. Digital Enterprise Research Institute, Galway, IEEE Internet Computing pp.87-90, 2007. **7.** *F. J. Carter Jr., T. Jambulingam, V. K. Gupta, N. Melone.* Technological innovations: A framework for communicating diffusion effects. Information & Management, 38(5), pp.277-287, 2001. **8.** *B. Celen, S. Kariv.* Observational learning under imperfect information. Games and Economic Behavior 47(1), pp.72-86, 2004. **9.** *A. Colman.* A Dictionary of Psychology. Originally published by Oxford University Press, 2001. **10.** *P. S. Dodds, R. Muhamad, D. J. Watts.* An Experimental Study of Search in Global Social Networks. Science, 8 August 2003, Vol. 301. no. 5634, pp.827-829, 2003. **11.** *L. Downes, C. Mui*. Unleashing the killer app: digital strategies for market dominance. Harvard Business School Press, 1998. **12.** *P. F. Drucker*. The New Realities. Transaction Publishers, Rev. Ed., 2003. **13.** *J. Fenn, A. Linden.* Gartner's Hype Cycle Special Report for 2005. 10th anniversary of Gartner's Hype Cycles, ID Number: G00130115, www.gartner.com/resources/130100/130115/gartners_ ype_c.pdf, 2005. **14.** *Gartner.* Understand Hype Cycle. Gartner Group, www.gartner.com/pages/story.php.id.8795.s.8.jsp, 2007. **15.** *M. Gladwell.* The Tipping Point: How Little Things Can Make a Big Difference. Little Brown, 2001. **16.** *M. Granovetter.* Threshold Models of Collective Behavior. The American Journal of Sociology, Vol. 83, No. 6, pp.1420-1443, 1978. **17.** *D. Gruhl, R. Guha, D. Liben-Nowell, A. Tomkins.* Information Diffusion Through Blogspace. In proceedings of the 13th International World Wide Web Conference (WWW'04), pp.491-501, 2004. **18.** *K.H. Lee.* Viral Architectures. MIT Media Lab, Viral Working Group, web.media.mit.edu/~kwan/ Projects/viralarchitectures.pdf, 2005. **19.** *E. Lesser, M. A. Fontaine, J. A. Slusher.* Knowledge and Communities. Elsevier LTD, Oxford, 2000. **20.** *C. P. Kindleberger.* Manias, Panics, and Crashes: A History of Financial Crises. Basic Books, New York, 1978. **21.** C. MacKay. Extraordinary Popular Delusions and the Madness of Crowds. With a foreword by Andrew Tobias, Harmony Books, New York, 1980. **22.** *D. Kempe, J. Kleinberg, E. Tardos.* Maximizing the Spread of Influence through a Social Network. SIGKDD '03, Washington, Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp.137-146, 2003. **23.** *S. Moscovici, E. Lage, M. Naffrenchoux.* Influences of a consistent minority on the responses of a majority in a colour perception task. Sociometry, Vol. 32, pp.365-80, 1969. **24.** *T. O'Reilly.* What Is Web 2.0 – Design Patterns and Business Models for the Next Generation of Software. Oreillynet.com, www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html, 2005. **25.** *T. Postmes, S. Brunsting.* Collective action in the age of the Internet: mass communication and online mobilization. Social Science Computer Review, Volume 20, Issue 3, Special issue: Psychology and the Internet, pp.290-301, 2002. **26.** *E. Rogers.* Diffusion of Innovations. Fifth Edition. Free Press, New York, 2003. **27.** *M. Rolfe.* Social networks and threshold models of collective behavior. University of Chicago, Workingpaper, December 10, 2004. **28.** *C.Russ.* Online Crowds – Extraordinary mass behavior on the Internet. Proceedings of i-Media '07, Graz, Austria, pp.65-76, 2007. **29.** *T. C. Schelling.* Micromotives and Macrobehavior. Norton, W. W. & Company, Inc, 1978. **30.** *R. J. Shiller.* Irrational Exuberance. University Presses of CA, 2nd ed., 2005. **31.** *D. J. Watts.* The "New" Science of Networks. Annual Review of Sociology Vol. 30, pp.243-270, 2004.

Поступила в редколлегию 15.02.08

UDC 681.3.06

V.A. SHEKHOVTSOV, NTU "KhPI", Kharkiv, Ukraine,
Ch. KOP, Alpine-Adriatic University of Klagenfurt, Austria
H.C. MAYR, Alpine-Adriatic University of Klagenfurt, Austria

TOWARDS QUALITY-AWARE PREDESIGN MODEL

У статті розглядаються основи пілходу до збирання семантики вимог якості у проміжну предпроектну модель. Цей підхід є поєднанням технологій Клагенфуртського концептуального предпроектування та аспектного предпроектування. Запропоновані додатки дозволяють включити до моделі ієрархію характеристик якості та подати наскрізні відношини між інтересами якості та основною функціональністю системи. Обговорені деякі напрямки інтеграції запропонованої моделі у процесс розробки програмного забезпечення, що керується якістю.

An approach to capturing the semantics of quality requirements into an intermediate predesign model is outlined. This approach combines Klagenfurt Conceptual Predesign and Aspectual Predesign techniques. Proposed extensions incorporate the hierarchy of quality characteristics into the predesign model and represent crosscutting relationships between the quality concerns and the main functionality of the system. Some directions of integration of the proposed model into quality-driven software process are discussed.

1. Introduction. One of the problems arising while developing an approach to incorporate quality-related issues into software process is a problem of finding an adequate representation of the semantics of quality requirements before performing design-time activities.

Following Klagenfurt Conceptual Predesign [12-13, 15] and Aspectual Predesign [19-20] approaches, to solve the above problem we propose to establish an intermediate semantic model (*predesign model*) residing between quality requirements elicitation and conceptual design. Such model has to describe the notion of the software quality that can be used on different stages of the software process, and capture the quality requirements semantics in a way that can be easily understood and verified by the system users. We call this model *Quality-Aware Predesign Model* (QAPM). In this paper we outline the main concepts of this model, more detailed description will be included in the follow-up papers.

The rest of the paper is organized as follows. Section 2 gives some important background information about software quality and existing predesign approaches. Section 3 describes the main features of the proposed predesign model. Section 4 is devoted to the integration of the described technique into broader context of quality-driven software process. Section 5 concludes the paper and shows the directions for future research.

2. Background. In this section, some necessary background information will be introduced.

2.1. Quality Models and Quality Requirements. To be able to describe the approach for analysis of quality-related information, it is necessary to select the notion of the software quality first. In this paper, we limit ourselves to the taxonomy approach to representing the product quality [7]. In this approach, quality is conceptualized as a hierarchy of quality attributes, with top-level attributes representing general quality characteristics (like functionality and reliability), whereas bottom-level attributes (quality sub-characteristics) representing more concrete characteristics (e.g. reliability can be decomposed into fault tolerance and availability). This representation forms the foundation for the quality model [2, 5-6]. There are many quality models proposed in literature (starting from [1]) and standardized by respective bodies such as ISO [9].

Following [8, 21], we assume that quality sub-characteristics are quantified via quality measures (indicators). For example, according to [21], "time behavior" sub-characteristic can be quantified via turnaround time, response time, CPU elapsed time, I/O processing time and several other indicators.

Indicators form the foundations for quality criteria and quality requirements. Every criterion reflects "a single aspect of quality of the system" [9]. According to [5-6] criteria can be seen as quality indicators connected to the particular system artifacts or its operations, e.g. for "response time" quality indicator the criteria can be "*response time for searching the customer by name*", "*response time for bank account withdrawal*" etc. Quality criteria together with threshold values form the quality requirements. For example, the requirement based on described criteria could look like this: "*response time of searching the customer by name must not exceed 1 second*".

Glinz [8] proposed classification of quality requirements introducing the concerns (matters of interest in a system) [4], in particular (a) functional concerns related to expected system functionality and (b) quality concerns related to quality characteristics defined by some quality model. After that, the set of requirements was decomposed into three main categories: (1) *functional requirements* related to functional concerns; (2) *quality requirements* related to quality concerns; (3) *constraints* constraining the solution space beyond what is necessary to meet the particular functional or quality requirement. We plan to base our model on this classification.

2.2. Handling Quality Requirements in Predesign Approaches. In this section, we will outline the approaches to handling the quality requirements in existing predesign approaches.

Klagenfurt Conceptual Predesign Model (KCPM) [12-13, 15] consists of a small set of semantic concepts such as *thing-type* (generalization of class and value type), *connection-type* (representing relationships between thing-types), or *operation-type* (modeling functional services). In this paper, we restrict ourselves to its tabular representation using glossaries. Though this model is built to capture the semantics of all kinds of requirements, non-functional requirements treatment is limited by collecting them in the *constraint glossary*. Each constraint (e.g. *The System shall process a minimum of 8 transactions per second*) could be related to at least one constraint type. In [12] a constraint type was a classification of the non-functional requirement (e.g. *performance requirement*). Since different kinds of classifications exist the constraint type was connected to one constraint category (e.g. "*IEEE Std. 830-1993"*). This way of connecting constraint categories and constraint types to constraint gave the designer more flexibility. He/she was allowed to define any kind of category ("*IEEE Std. 830-1993"*, "*My Characteristics"* etc). Within this category it was possible to collect the types of requirements which belong to it. Once the types were defined the designer was able to relate the collected constraints to one or more constraint types related to different categories.

The main goal of an *Aspectual Predesign* technique [19-20] was extending the KCPM to make it able to deal with crosscutting concerns in the problem space. It aimed at capturing the semantics of "aspectual" (crosscutting) requirements as defined by aspect-oriented software development (AOSD) terminology [4] into the predesign model (Aspectual Predesign Model, APM) similar in its purpose to KCPM. In this model, thing-types are used to represent concerns in AOSD sense, crosscutting behavior units implementing quality requirements (advices, interceptors) are represented via operation-types; pointcuts (rules that connect advices to some places in model where they are supposed to be called) are represented via modified connection-types. Aspectual predesign can be seen both as an extension to the Klagenfurt conceptual predesign that allows mapping the aspectual requirements and as an intermediate step of the AOSD residing between aspect-oriented requirements engineering and aspect-oriented modeling.

The two predesign approaches are complementary. Whereas KCPM represents quality requirements as constraints and allows user-supplied classification of these requirements, APM allows treating the requirements as belonging to crosscutting concerns and offers some guidance in separation of these concerns. It seems feasible to merge these approaches in a way that makes the resulting technique benefit from their advantages. The outline of the possible results of this merge is presented in the following section.

3. Outline of the Model. Several problems need to be solved during the model development: (1) allowing integration of the quality model; (2) extending the KCPM metamodel to integrate complete representation of quality requirements; (3) implementing the semantic support for separation of quality-related and functional concerns; (4) implementing support for relationships between these concerns. In this section, we describe our approach to resolving these problems.

3.1. Integrating Quality Models into QAPM. For allowing a flexible integration of the quality–related information, we introduce two new semantic concepts for our predesign model: a *quality characteristic* and a *quality model*. We cannot use existing concepts (such as thing types) for this purpose because they represent types of things whereas quality characteristics are instances of the particular high-level concept. A quality characteristic is a semantic concept for elements from all levels of a quality model hierarchy; quality model represents the particular instance of this hierarchy. For quality indicators, their units of measurement are values for "value domain" meta-attribute of the quality characteristic.

Fig.1 contains the fragment of a quality model glossary corresponding to the extended ISO 9126 quality model (in particular, the "Efficiency" high-level quality characteristic).

id#	Name	belongs to	value domain
Q04	Efficiency		
Q04-1	Time behavior	Q04, Efficiency	
Q04-1-1	Response time	Q04-1, Time behavior	seconds

Figure 1: Part of the quality model glossary corresponding to efficiency

The reason of storing all the quality model information in the glossary reflects the "mixed model" paradigm of the quality model construction [5] making possible to tailor already existing quality models for the particular problems. In our model, analysts can add or modify quality characteristics and indicators of different nature.

3.2. Modeling Crosscutting Concerns and Requirements. In this section, we show how our model allows capturing crosscutting concerns and requirements.

In our model, quality characteristics and sub-characteristics are treated (following [8, 16]) as *concerns*. We also follow [16] in distinguishing the dominant functional concern which controls the decomposition of the system and modeling all other concerns (in particular, all quality concerns) as crosscutting concerns. As the quality concerns form the primary interest of this paper, we

assume that the dominant concern is the main functionality of the system. This concern defines the decomposition of the predesign model into the set of thing-types and other KCPM artifacts.

In this paper, we also assume that quality concerns directly correspond to the quality characteristics and sub-characteristics in the accepted quality model (e.g. for ISO 9126 quality model the candidate concerns are "Efficiency", "Usability", "Time behavior" etc.) As a result, we do not need any special notation to represent these concerns; the quality model glossary will serve the purpose of quality concern glossary as well. If some quality characteristics are of no interest to the current system, they can be simply ignored in the rest of the model.

To be able to represent crosscutting relationships between functional and quality concerns, we need to decide on a *join point model* [6] based on captured requirements semantics. This model defines the set of all possible places where the functionality of the base concern can be extended or replaced with the functionality implementing the crosscutting concern, and, on the other hand, the set of all possible model artifacts belonging to the crosscutting concern that can be chosen for this extension. The join points in this model are *thing-type, connection-type, operation-type* and *cooperation-type*.

After the join point model is defined, the next step is to establish the semantics of quality requirements. We propose *a constraint* to be a semantic concept corresponding to the quality requirement. To reflect the relationship between base and quality concerns every such constraint will contain the references to particular quality concern (quality characteristic) and the element of dominant functional concern belonging to the joint point model (KCPM artifact). The QAPM metamodel of quality requirement is shown on Fig.2.

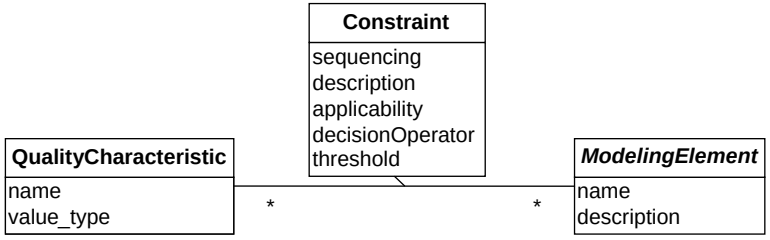

Figure 2: Part of QAPM metamodel describing the quality requirement as a constraint

The *ModelingElement* is the root of the schema elements hierarchy in the KCPM metamodel. We decided to associate the *QualityCharacteristic* to this abstract meta-class and enhance this association using the *Constraint* associa-

tive meta-class. The meta-attributes characterizing the quality requirement constraint are as follows:

1. "*sequencing*" reflects the temporal and conditional dependencies between base and quality concern elements [3]. The set of possible values reflecting these dependencies includes "before", "after", "wrap", "instead", "concurrently", "if", and "if not".
2. *"applicability"* represents applicability condition for this requirement (e.g. "during peak hours", "during startup and shutdown", "if the system is in the safe mode" etc.)
3. *"description"* contains the description of the requirement. For imprecise requirements, this meta-attribute is supposed to contain all the information available for the requirement, e.g. "the system must be secure". For refined requirements, the following two meta-attributes will be used as well.
4. *"decisionOperator"* contains the operator which needs to be applied to the threshold value to determine if the requirement is satisfied or not (e.g. "equals", "less" or more complicated operators)
5. *"threshold"* contains the threshold value.

Fig.3 shows the fragment of a constraint glossary representing quality requirement. We suppose thing-type *Order* and cooperation-type *Order department checks articles* are already defined in a model.

id#	quality characteristic	functional element	sequencing	description	applicability	decision operator	threshold
C02	Q04-1-1, Response time	E01, Order department checks articles	wrap	the response time must be short	during peak hours	less	0.5

Figure 3: Quality requirements in the QAPM constraint glossary

4. Model Integration. The QAPM is supposed to be integrated into wider context of *Quality-Driven Software Process* – specialized software process aimed at integrating quality into all the stages of the software development from requirements elicitation to code generation. In this section, we take a detailed look at various aspects of this integration.

4.1. QAPM Information Suppliers. Our model obtains its input information from the requirements elicitation stage of the software process. In the framework of the quality-driven software process, we plan to support this stage (Quality Requirements Elicitation) with some special techniques.

For example, when natural language requirements specifications are available and the stakeholders trust them, it is possible to elicit quality requirements from these specifications using NLP algorithms and transfer their semantics into QAPM glossaries ([14] represents a preliminary technique aimed at this goal).

Another technique is supposed to be used if the formal requirements specifications are difficult to obtain or cannot be completely trusted. The proposed approach collects the stakeholders experience allowing them to assess the qualities of the system under development interactively in context of its usage processes. To achieve this, it is planned to construct a special tool [18] implementing an interactive simulation environment. In this environment, stakeholders can experience the qualities of the system under development in context of the usage processes carried out in their organizations and make assessments of these qualities. These assessments will serve as sources of the quality requirements. The semantics of the requirements elicited via this environment can also be transferred into QAPM glossaries.

4.2. QAPM Information Consumers. Information represented in QAPM glossaries is planned to serve as a source for two other steps of the Quality-Driven Software Process: Quality-Driven Architecture Design and Quality-Driven Code Generation.

For the architecture design step of the software process it is planned to implement a tool for creating architecture of the system under development that entails that system to have the required qualities. This problem is broken down into the set of problems related to selecting a software architecture artifact which possesses the desired qualities (artifact selection) while reaching an agreement between the desired system quality and the resource constraints (artifact negotiation). Paper [10] describes a technique supporting the quality-based selection of specific development artifacts (BPM methodologies). For quality-driven software process, it will be generalized to cover all the software artifacts. We plan to obtain the information about the desired qualities controlling the artifact selection and negotiation from QAPM quality model.

For the code generation step it is planned to implement a tool for creating the code of the system under development in a way that complies with the architecture worked out earlier. To achieve this goal, it is planned to utilize the power of modern code-generation techniques (such as OO-Method [17]) by finding the way to adapt quality-related information so it can influence some aspects of the code generation. To be able to perform actual quality-driven code generation, it is planned to integrate the concepts related to quality into OO-Method (its notation, methodology, and abstract execution model). For initial

representation of quality-related information, we plan to use QAPM enhancing the approach from [11], which suggests using original KCPM for this purpose.

5. Conclusions and Future Work. In this paper, we outlined the basic concepts of a quality-aware intermediate model allowing capturing quality requirements semantics into glossary entries that can be verified by the end users. This model can serve as a source for information necessary on other stages of the quality-driven software process. In fact, we plan to use this model as core "scratch pad" of the problem domain for this process. In future, we plan to implement a software tool supporting this model, as well as perform all the described actions necessary to integrate QAPM into the software process.

References. 1. *Boehm, B.W., Brown, J.R., et al.* Characteristics of Software Quality // Vol.1 of TRW Series of Software Technology. North-Holland, 1978. **2.** *Carvallo, J., Franch, X., Grau, G., Quer, C.* Reaching an Agreement on COTS Quality through the Use of Quality Models // Proc. ICSE 2nd Workshop on Software Quality, 2004. **3.** *Chitchyan, R., Rashid, A., et al,* Semantics-based Composition for Aspect-Oriented Requirements Engineering // Proc. AOSD'07, ACM, 2007. **4.** *Filman, R., Elrad, T., Clarke, S., Aksit, M.* Aspect-Oriented Software Development. Addison-Wesley, 2006. **5.** *Firesmith, D.* Quality Requirements Checklist // J. of Object Technology, 4:9, 2005, p.31-38. **6.** *Firesmith, D.* Using Quality Models to Engineer Quality Requirements // J. of Object Technology, 2:5, 2003, p.67-75. **7.** *Garvin, D.A.* Competing on the Eight Dimensions of Quality // Harvard Business Review, 65:6, 1987, p.101-109. **8.** *Glinz, M.* On Non-Functional Requirements // Proc. RE'07, IEEE, 2007. **9.** ISO/IEC 9126-1, Software Engineering – Product Quality – Part 1: Quality model, 2001. **10.** *Kaschek, R., Pavlov, R., Shekhovtsov, V., Zlatkin, S.* Characterization and tool supported selection of business process modeling methodologies // Technologies for Business Information Systems. Springer, 2007, p.25-37. **11.** *Kop, C, Mayr, H. C., Yevdoshenko, N.*: Requirements Modeling and MDA. Proposal for a Combined Approach // Proc ISD 2006, Springer, 2007 **12.** *Kop, Ch.* Rechnergestützte Katalogisierung von Anforderungspezifikationen und deren Transformation in ein konzeptuelles Modell. Doctoral thesis, University Klagenfurt, 2002. **13.** *Kop, Ch., Mayr, H.C.* Mapping Functional Requirements: From Natural Language to Conceptual Schemata // Proc. SEA'02, 2002, p.82-87. **14.** *Kostanyan, A., Shekhovtsov, V.* Towards Entropy-Based Requirements Elicitation // Proc. ISTA'2007, LNI P-107, GI-Edition, 2007, p.105-116. **15.** *Mayr, H.C.; Kop, Ch.*: A User Centered Approach to Requirements Modeling // Proc. Modellierung 2002. LNI P-12, GI-Edition, 2002, p.75-86. **16.** *Meier, S., Reinhard, T., Seybold, C., Glinz, M.* Aspect-Oriented Modeling with Integrated Object Models // Proc. Modellierung 2006, GI-Edition, 2006, p.129-144. **17.** *Pastor, O., Gómez, J., Insfrán, E., Pelechano, V.*: The OO-Method approach for information systems modeling: from object-oriented conceptual modeling to automated programming // Information Systems, 24:7, 2001, p.507-534. **18.** *Shekhovtsov, V., Kaschek, R., Zlatkin, S.* Constructing POSE: a Tool for Eliciting Quality Requirements // Proc. ISTA'2007, LNI P-107, GI-Edition, 2007, p.187-199. **19.** *Shekhovtsov, V., Kostanyan, A.* Aspectual Predesign // Proc. ISTA'2005, LNI P-63, GI-Edition, 2005, p.216-226. **20.** *Shekhovtsov, V., Kostanyan, A., Gritskov, E., Litvinenko, Y.* Tool Supported Aspectual Predesign // Proc. ISTA'2006, LNI P-84, GI-Edition, 2006, p.153-164. **21.** *van Zeist, B., Hendriks, P., Paulussen R., Trienekens, J.* Quality of Software Products. Experiences with a quality model. Kluwer , 1996.

Поступила в редколлегию 20.02.08

UDC 512.086

M.TALIB, Ph.D., Department of Computer Science,
University of Botswana,
A.ABUSUKHON, M.Sc., School of Computing and Technology,
University of Sunderland

GRAPHICS TECHNOLOGY TO MODEL THE PROBLEMS OF CALCULUS USING ANALYTICAL GEOMETRY

Ця стаття містить опис деяких методів візуалізації, що дають можливість використовувати технології комп'ютерної графіки для моделювання задач математичного аналізу та аналітичної геометрії. Детально розглядаються питання використання комп'ютерних зображень та робиться огляд використання комп'ютерної анімації для цієї візуалізації.

The paper contains some general background and some of the visualization methods that have been used to bring computer graphics technology to model mathematical problems of Calculus with Analytical Geometry. Computer-generated images have been length and breath of the paper as a source of additional background information on visual mathematics and an overview of selected animations concerned with mathematical visualization.

1. Introduction. The intention of this paper is to show the natural interrelationship between calculus mathematics and computer graphics. This article will concentrate for the most part in IT perspective on the progress, techniques, and prospects of mathematical visualization, emphasizing those areas of 2D and 3D geometry where interactive paradigms are of growing importance. [1]

Due to substantial changes that technology has brought in the recent years, instruction in mathematics will have to catch up with the new circumstances or else become increasingly irrelevant. With added pressure from rapid development in Multimedia, it is even more demanding to train our students to think clearly, critically, constructively, and creatively about problems they might encounter in real world. It is our job to help students to gain the ability to use mathematical methods and tools whenever they seem appropriate and helpful. Computer oriented mathematics courses should focus more on cooperative learning, problem solving, and investigative learning as an important part of education.

2. Computer Algebraic System. In the mid-eighties the availability of CAS for personal computers attracted mathematics educators to the possibility of using them in the classroom. CAS technology with its powerful combination of numeric and symbolic computation, colorful 2D & 3D graphics as in figure 1 and